Characterizing timing and memory-requirements of the $\mathcal{F}$-statistic implementations in LALSuite

Reinhard Prix[*]
Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut, D-30167 Hannover, Germany
(Dated: 2017-07-18 17:50:43 +0200; commitID: 1c97c53-CLEAN; LIGO-T1600531-v4+)

We develop a generic (method-independent) timing model for the $\mathcal{F}$-statistic computation. We present and quantify method-specific timing and memory models for the Resampling-FFT method, as well as the (older) Demod method. The timing models are found to agree well with the measured CPU times with a standard deviation on the error of less than 10%.

I. THE $\mathcal{F}$-STATISTIC TIMING MODEL

In the following we denote as $\mathcal{T}$ any times (per detector) referring to a single call to `XLALComputeFstat()`, which computes the $\mathcal{F}$-statistic over a vector of N_{Fbin} frequency bins, τ any times per detector per output $\mathcal{F}$-statistic frequency bin N_{Fbin}, and $\tau^{(0)}$ any "fundamental" timing coefficients that don't scale with any of the algorithm parameters (at least approximately), and which should only depend on the hardware and code optimization settings.

The general $\mathcal{F}$-statistic timing model can be expressed in terms of two contributions:

1. the average *core* time $\tau_{\mathcal{F}}^{\mathrm{core}}$ to compute a single $\mathcal{F}$-statistic frequency bin for one detector, excluding the time to compute any quantities that are stored in the buffer, and

2. the average time $\tau_{\mathcal{F}}^{\mathrm{buffer}}$ per frequency bin per detector to (re-)compute all buffered quantities

We can write the *effective* average $\mathcal{F}$-statistic time $\tau_{\mathcal{F}}^{\mathrm{eff}}$ per template per detector as

$$\tau_{\mathcal{F}}^{\mathrm{eff}} = \tau_{\mathcal{F}}^{\mathrm{core}} + b\,\tau_{\mathcal{F}}^{\mathrm{buffer}}\,, \tag{1}$$

where the "buffer miss fraction" $b \in [0,1]$ quantifies how many times the buffer needed to be recomputed per call to `XLALComputeFstat()` per output frequency bin, i.e. $b = 0$ is 100% efficient buffering, i.e. the buffer never needed to be re-computed, while $b = 1$ means the buffer needed to be re-computed for every call to `XLALComputeFstat()`. The buffer contains sky-dependent quantities (antenna-pattern functions, SSB time delays, SSB-resampled time-series, ...) as well as quantities that depend on the binary-orbital parameters (such as binary-orbital time delays and corresponding resampled time-series). Typically we can re-use the buffer for all spindowns for given sky-position and binary-orbital parameters, and so in most cases we will have

$$b = \frac{1}{N_{\{\dot{f},\ddot{f},\ldots\}}}\,, \tag{2}$$

in terms of the number of spindown templates of all orders computed per sky-position and binary-orbital parameters.

The timing collection within the $\mathcal{F}$-statistic code measures and outputs these quantities as

$$\begin{aligned}
\texttt{tauF\_eff} &\equiv \tau_{\mathcal{F}}^{\mathrm{eff}} = \tau_{\mathcal{F}}^{\mathrm{core}} + b\,\tau_{\mathcal{F}}^{\mathrm{buffer}}\,, && (3)\\
\texttt{tauF\_core} &\equiv \tau_{\mathcal{F}}^{\mathrm{core}}\,, && (4)\\
\texttt{tauF\_buffer} &\equiv \tau_{\mathcal{F}}^{\mathrm{buffer}}\,, && (5)\\
\texttt{bufferMissFrac} &\equiv b\,. && (6)
\end{aligned}$$

A *timing model* consists of expressing the $\mathcal{F}$-statistic times $\tau_{\mathcal{F}}^{\mathrm{core}}$ and $\tau_{\mathcal{F}}^{\mathrm{buffer}}$ in terms of algorithm parameters and fundamental timing coefficients $\tau_{\ldots}^{(0)}$, which allows us to predict $\mathcal{F}$-statistic compute times as a function of algorithm parameters, assuming we have measured the fundamental timing coefficients for a given code and hardware.

[*] reinhard.prix@aei.mpg.de

A. Resampling timing model

1. Notation

There are essentially four different vector lengths over which various operations are performed in the Resampling-FFT $\mathcal{F}$-statistic code:

$N_{\rm Fbin}$: The user-requested number of output frequency bins, with spacing df, for which the $\mathcal{F}$-statistic is computed.

$N^{\rm DET}_{\rm samp}$: The (maximum over detectors) number of samples in the input timeseries (in the detector frame.

$N^{\rm SRC}_{\rm samp}$: The (maximum over detectors) number of samples of the input timeseries, interpolated into the source-frame.

$N^{\rm FFT}_{\rm samp}$: The length of the (zero-padded) timeseries on which the FFT is performed.

which are given by the following relations:

The sampling step $dt_{\rm DET}$ of the input timeseries in the detector frame (DET) is given in terms of the frequency bandwidth $\Delta f_{\rm sft}$ of the input SFTs, namely

$$dt_{\rm DET} \equiv \frac{1}{\Delta f_{\rm sft}}\,, \tag{7}$$

and the number of detector-frame time-samples are therefore

$$N^{\rm DET}_{\rm samp} = \frac{T_{\rm coh}}{dt_{\rm DET}} = T_{\rm coh}\,\Delta f_{\rm sft}\,, \tag{8}$$

where $T_{\rm coh}$ is the (maximum over IFOs) maximum coherent data span, which is determined by the actual SFTs used and can therefore generally be less than the user-requested (e.g. via segment list) span. The duration $T_{\rm FFT}$ of the final timeseries to be FFT'ed is determined by the user-requested frequency resolution df, namely

$$T_{\rm FFT} = \frac{D}{df}\,, \quad \text{with} \quad D \equiv \lceil T_{\rm coh}\, df \rceil\,, \tag{9}$$

where $D \in \mathbb{N}^+$ ensures that we always have $T_{\rm FFT} \geq T_{\rm coh}$. A choice $D > 1$ is required when the user asked for a coarse frequency resolution $df > 1/T_{\rm coh}$ in order to still use all the data (i.e. we always use zero-padding rather than truncating the input timeseries) and therefore not lose SNR. In the regular case of a frequency resolution $df < 1/T_{\rm coh}$ we have $D = 1$. The *decimation* factor D amounts to internally using a finer frequency resolution

$$df_{\rm internal} \equiv \frac{df}{D} \leq \frac{1}{T_{\rm coh}}\,, \tag{10}$$

and then returning only the user-requested frequency bins with resolution df, namely by returning only every D'th frequency bin.

With the original detector-frame sampling step $dt_{\rm DET}$ this would correspond to a number of time samples to be FFT'ed:

$$N^{\rm FFT(0)}_{\rm samp} = \frac{T_{\rm FFT}}{dt_{\rm DET}} = \frac{\Delta f_{\rm sft}}{df_{\rm internal}}\,. \tag{11}$$

However, in order to ensure the most efficient (and most consistent) FFT performance, we round the number of FFT bins up to the next power of 2, namely

$$N^{\rm FFT}_{\rm samp} = 2^{\lceil \log_2 N^{\rm FFT(0)}_{\rm samp} \rceil}\,, \tag{12}$$

which is achieved by decreasing the sampling step (i.e. increasing the frequency band) of the resampled timeseries in the source frame (SRC), i.e.

$$dt_{\rm SRC} \equiv \frac{T_{\rm FFT}}{N^{\rm FFT}_{\rm samp}}\,. \tag{13}$$

The (longest) coherent detector-frame time-series of span $T_{\rm coh}$ interpolated into the source-frame will therefore have $N^{\rm SRC}_{\rm samp}$ samples, given as

$$N^{\rm SRC}_{\rm samp} \equiv \frac{T_{\rm coh}}{dt_{\rm SRC}} = \mathcal{R}\, N^{\rm FFT}_{\rm samp}\,, \tag{14}$$

where we defined the (internal) frequency resolution $\mathcal{R} \le 1$ in "natural" units $1/T_{\rm coh}$ as

$$df_{\rm internal} = \mathcal{R}\,\frac{1}{T_{\rm coh}}\,, \tag{15}$$

namely

$$\mathcal{R} \equiv \frac{T_{\rm coh}}{T_{\rm FFT}} = T_{\rm coh}\; df_{\rm internal} \le 1\,. \tag{16}$$

From Eq. (8) and Eq. (11) we find a similar relation to Eq. (14) for $N^{\rm DET}_{\rm samp}$, namely

$$N^{\rm DET}_{\rm samp} = \mathcal{R}\, N^{\rm FFT(0)}_{\rm samp}\,. \tag{17}$$

B. Required SFT frequency band $\Delta f_{\rm sft}$

When the caller requests a physical SRC-frame frequency band Δf to be analyzed, we need to expand this to $\Delta f_{\rm load}$ when loading the data from SFTs: this wider band contains an extra "frequency drift" sideband $\Delta f_{\rm drift}$ to account for Doppler shifts and spindowns (computed from `XLALCWSignalCoveringBand()` [1]), and an additional 16 extra SFT bins (8 on either side[1]) added to use similar noise bins as Demod near the SFT edges, i.e.

$$\Delta f_{\rm load} = \Delta f + \Delta f_{\rm drift} + \frac{16}{T_{\rm sft}}\,, \tag{18}$$

$$\Delta f_{\rm drift} \approx 2.12\times 10^{-4}\left(f_{\rm max} + |\dot{f}|_{\rm max}\,\frac{T_{\rm span}}{2} + |\ddot{f}|_{\rm max}\,\frac{T^2_{\rm span}}{8} + \dots\right)\,, \tag{19}$$

where $T_{\rm span}$ denotes the total duration of data included in this calculation. The handling of this can differ between semi-coherent search codes, for example for the GCT code, $T_{\rm span}$ denotes the *total* span of data over all semi-coherent segments, whereas for the Weave code, this could be computed (and data loaded) on a per-segment basis, such that for Weave $T_{\rm span} = T_{\rm coh}$. An additional "transition band" is added internally in order to allow for the roll-off from the Hamming-windowed sinc-interpolation used in Barycentric resampling, which further increases this to

$$\Delta f_{\rm sft} = \Delta f_{\rm load}\left(1 + \frac{4}{2\,{\rm Dterms}+1}\right)\,, \tag{20}$$

where Dterms is the (user-specified) number of sinc-kernel terms to use (on either side, therefore the window-size is $2\,{\rm Dterms}+1$) in the barycentric resampling interpolation.

Note that these expressions (especially $\Delta f_{\rm drift}$) are only approximations of what is computed in the F-stat codes, and due to the power-of-two rounding of Eq. (12), this can in some cases lead to relatively large deviations from the actual measured runtimes. For the most accurate and reliable prediction use the `octapps` function `predictFstatTimeAndMemory()` [2].

The "physical" number of maximal output frequency bins is

$$N^{\rm max}_{\rm Fbin} = \frac{\Delta f}{df}\,, \tag{21}$$

while calls to `XLALComputeFstat()` can request to compute fewer output frequency bins $N_{\rm Fbin} \le N^{\rm max}_{\rm Fbin}$, provided they fall into the requested frequency band Δf. Note that for semi-coherent codes such as the GCT code, Δf will not be just the user-requested search frequency band Δf_0, but will typically include additional sideband bins, for example in the case of the GCT code we'll have a wider band of

$$\Delta f = \Delta f_0 + \frac{1}{2}\left(T_{\rm span}\, d\dot{f} + T^2_{\rm span}\, d\ddot{f}\right)\,, \tag{22}$$

where the extra sideband is referred to as `extraBinsFstat` in the GCT code.

[1] called `extraBinsMethod` in the code

1. Timing model contributions

We can break the time to compute the Resampling $\mathcal{F}$-statistic into the following contributions: the *core* compute time $\mathcal{T}_{\mathcal{F}}^{\mathrm{core}}$ has the following contributions $\mathcal{T}_{\mathcal{F}}^{\mathrm{core}} = \mathcal{T}_{\mathrm{Fbin}} + \mathcal{T}_{\mathrm{spin}} + \mathcal{T}_{\mathrm{FFT}}$, which scale differently with search parameters, namely

1. Apply spindown-correction and frequency shift to source-frame timeseries, which scales with $N_{\mathrm{samp}}^{\mathrm{SRC}}$, and therefore
$$\mathcal{T}_{\mathrm{spin}} = N_{\mathrm{samp}}^{\mathrm{SRC}}\,\tau_{\mathrm{spin}}^{(0)} = \mathcal{R}\,N_{\mathrm{samp}}^{\mathrm{FFT}}\,\tau_{\mathrm{spin}}^{(0)}\,. \tag{23}$$

2. Apply the FFT, which operates on the zero-padded timeseries of length $N_{\mathrm{samp}}^{\mathrm{FFT}}$, therefore
$$\mathcal{T}_{\mathrm{FFT}} = 5\,N_{\mathrm{samp}}^{\mathrm{FFT}}\,\log_2(N_{\mathrm{samp}}^{\mathrm{FFT}})\,\tau_{\mathrm{FFT}}^{(0)}\,, \tag{24}$$
which was defined in such a way that we can directly the FFT timing coefficient $\tau_{\mathrm{FFT}}^{(0)}$ to the benchmark "speed" definition of FFTW in `http://www.fftw.org/speed/`. This is defined in terms of "mega-flops" as mflops $\equiv 5\,N_{\mathrm{samp}}^{\mathrm{FFT}}\log_2(N_{\mathrm{samp}}^{\mathrm{FFT}})/(\mathcal{T}_{\mathrm{FFT}}\,10^6)$, and so FFTW's "flops" are simply
$$\mathrm{FFTflops} = \frac{1}{\tau_{\mathrm{FFT}}^{(0)}}\,. \tag{25}$$
The FFT timing coefficient $\tau_{\mathrm{FFT}}^{(0)}$ is found to be relatively constant for FFTs of length $N_{\mathrm{samp}}^{\mathrm{FFT}} > 2^{18} = 262144$, while it is noticeably smaller but also more variable for shorter FFTs (in agreement with FFTW's benchmark results for single-precision complex FFTs on 64bit hardware shown here: `http://www.fftw.org/speed/CoreDuo-3.0GHz-icc64/`). In practice we therefore use two FFT timing coefficients, $\tau_{\mathrm{FFT}}^{(0)}(N \le 2^{18})$ and $\tau_{\mathrm{FFT}}^{(0)}(N > 2^{18})$.

3. various operations on the N_{Fbin} output bins, such as copying the bins, normalizing them, computing $\mathcal{F}$ from F_a, F_b and summing them over detectors $F_{a,b} = \sum_X F_{a,b}^X$ (here considering time per detector), which can be summarized as a time contribution per output frequency bin $\tau_{\mathrm{Fbin}}^{(0)}$, internally composed of these parts
$$\mathcal{T}_{\mathrm{Fbin}} = \mathcal{T}_{\mathrm{copy}} + \mathcal{T}_{\mathrm{norm}} + \mathcal{T}_{\mathrm{sumFabX}} + \mathcal{T}_{\mathrm{Fab2F}} = N_{\mathrm{Fbin}}\,\tau_{\mathrm{Fbin}}^{(0)}\,. \tag{26}$$

We therefore obtain the core resampling time per detector per output frequency bin N_{Fbin} as
$$\tau_{\mathcal{F}}^{\mathrm{core}} \equiv \frac{\mathcal{T}_{\mathcal{F}}^{\mathrm{core}}}{N_{\mathrm{Fbin}}} = \tau_{\mathrm{Fbin}}^{(0)} + \frac{N_{\mathrm{samp}}^{\mathrm{FFT}}}{N_{\mathrm{Fbin}}}\left[\mathcal{R}\,\tau_{\mathrm{spin}}^{(0)} + 5\log_2(N_{\mathrm{samp}}^{\mathrm{FFT}})\,\tau_{\mathrm{FFT}}^{(0)}\right]\,. \tag{27}$$

We also need to compute all the buffered quantities: antenna-patterns and source-frame time-delays, barycenter and interpolate the input detector-frame time-series into the source frame. This contribution contains some terms that scale with the number of input SFTs, and others that scale with the number of input samples of the SFTs (in case of gaps), as well as $N_{\mathrm{samp}}^{\mathrm{SRC}}$ and $N_{\mathrm{samp}}^{\mathrm{DET}}$. However, we postulate the following approximate scaling relation for the dominant contribution (barycentering) as
$$\mathcal{T}_{\mathcal{F}}^{\mathrm{buffer}} \sim \mathcal{T}_{\mathrm{bary}} \sim N_{\mathrm{samp}}^{\mathrm{SRC}}\,\tau_{\mathrm{bary}}^{(0)} = \mathcal{R}\,N_{\mathrm{samp}}^{\mathrm{FFT}}\,\tau_{\mathrm{bary}}^{(0)}\,, \tag{28}$$
and so
$$\tau_{\mathcal{F}}^{\mathrm{buffer}} \equiv \frac{\mathcal{T}_{\mathcal{F}}^{\mathrm{buffer}}}{N_{\mathrm{Fbin}}} = \mathcal{R}\,\frac{N_{\mathrm{samp}}^{\mathrm{FFT}}}{N_{\mathrm{Fbin}}}\,\tau_{\mathrm{bary}}^{(0)}\,. \tag{29}$$

The complete Resampling timing model for the effective average time per template per detector of Eq. (1) can therefore be expressed as
$$\tau_{\mathcal{F}}^{\mathrm{eff}} = \tau_{\mathrm{Fbin}}^{(0)} + \frac{N_{\mathrm{samp}}^{\mathrm{FFT}}}{N_{\mathrm{Fbin}}}\left[\mathcal{R}\left(\tau_{\mathrm{spin}}^{(0)} + b\,\tau_{\mathrm{bary}}^{(0)}\right) + 5\log_2(N_{\mathrm{samp}}^{\mathrm{FFT}})\,\tau_{\mathrm{FFT}}^{(0)}\right]\,. \tag{30}$$

C. Demod Timing Model

The timing model for the Demod implementation of the $\mathcal{F}$-statistic is much simpler. The core Demod $\mathcal{F}$-statistic time $\tau_{\mathcal{F}}^{\mathrm{core}}$ per detector is simply proportional to the number of SFTs N_{sft} from the single detector, i.e.

$$\tau_{\mathcal{F}}^{\mathrm{core}} = N_{\mathrm{sft}}\, \tau_{\mathrm{core,LD}}^{(0)} , \tag{31}$$

where $\tau_{\mathrm{core,LD}}^{(0)}$ is the fundamental timing coefficient expressing core Demod time *per SFT*.

In the Demod case, the time $\mathcal{T}_{\mathcal{F}}^{\mathrm{buffer}}$ to compute all the buffered quantities (sky-dependent antenna-patterns and sky- and binary-orbital dependent timings) for one detector is simply proportional to the number of SFTs from that detector, and we can obtain

$$\tau_{\mathcal{F}}^{\mathrm{buffer}} = \frac{N_{\mathrm{sft}}}{N_{\mathrm{Fbin}}}\, \tau_{\mathrm{buffer,LD}}^{(0)} , \tag{32}$$

in terms of a fundamental timing coefficient $\tau_{\mathrm{buffer,LD}}^{(0)}$, which denotes the time to compute all the buffered quantities for a single SFT. We can therefore write the full Demod timing model $\tau_{\mathcal{F}}^{\mathrm{eff}}$ for the time per template per detector as

$$\tau_{\mathcal{F}}^{\mathrm{eff}} = N_{\mathrm{sft}} \left[\tau_{\mathrm{core,LD}}^{(0)} + \frac{b}{N_{\mathrm{Fbin}}}\, \tau_{\mathrm{buffer,LD}}^{(0)} \right] , \tag{33}$$

in terms of the number of SFTs N_{sft} coherently analyzed from one detector.

II. MEMORY MODEL

In a similar way to the timing model we can enumerate the amount of memory required to perform the calculation of the $\mathcal{F}$-statistic.

A. Resampling memory model

- **Objects stored for each detector and each segment of a semi-coherent search:**
  - the original SFTs turned into a detector-frame `COMPLEX8` timeseries:
  $$\mathrm{mem}\left[\texttt{COMPLEX8TimeSeries-DET}\right] = N_{\mathrm{samp}}^{\mathrm{DET}} \times \mathrm{mem}\left[\texttt{C8}\right] .$$
  - two timeseries interpolated and barycentered into the source frame, multiplied by $a(t)$ or $b(t)$, respectively:
  $$\mathrm{mem}\left[\texttt{COMPLEX8TimeSeries-SRC-[a|b]}\right] = 2\, N_{\mathrm{samp}}^{\mathrm{SRC}} \times \mathrm{mem}\left[\texttt{C8}\right] .$$
  - the FFTW plan: we're *assuming* it should be roughly $N_{\mathrm{samp}}^{\mathrm{FFT}}$ `COMPLEX8` numbers(?)
  $$\mathrm{mem}\left[\texttt{FFT-plan}\right] \approx N_{\mathrm{samp}}^{\mathrm{FFT}} \times \mathrm{mem}\left[\texttt{C8}\right] .$$
- **Objects stored only once for all segments, using a shared "workspace":**
  - Two `COMPLEX8` vectors for temporary storage of SRC-frame timeseries:
  $$\mathrm{mem}\left[\texttt{TStmp[1|2]-SRC}\right] = 2 N_{\mathrm{samp}}^{\mathrm{SRC}} \times \mathrm{mem}\left[\texttt{C8}\right] ,$$
  - One `REAL8` timeseries holding time-differences between SRC and DET frames (only used for barycentering:
  $$\mathrm{mem}\left[\texttt{SRCtimes-DET}\right] = N_{\mathrm{samp}}^{\mathrm{SRC}} \times \mathrm{mem}\left[\texttt{R8}\right] .$$
  - the input and output vectors for the FFT (currently *not* using in-place transform):
  $$\mathrm{mem}\left[\texttt{TS-FFT + FabX-Raw}\right] = 2\, N_{\mathrm{samp}}^{\mathrm{FFT}} \times \mathrm{mem}\left[\texttt{C8}\right] ,$$

– temporary storage of $F^X_{a,b}$ and $F_{a,b}$:

$$\mathrm{mem}\left[\texttt{Fab}\right] = 4\,N_{\mathrm{Fbin}} \times \mathrm{mem}\left[\texttt{C8}\right] \,.$$

where $\mathrm{mem}\left[\texttt{C8}\right] = \mathrm{mem}\left[\texttt{R8}\right] = 8\,\mathrm{bytes}$ and $\mathrm{mem}\left[\texttt{R4}\right] = 4\,\mathrm{bytes}$.

We can therefore express the different memory blocks as

$$\mathrm{mem}\left[\texttt{ResampDataPerDetPerSeg}\right] = \left[\mathcal{R}\,N^{\mathrm{FFT}(0)}_{\mathrm{samp}} + (1+2\mathcal{R})\,N^{\mathrm{FFT}}_{\mathrm{samp}}\right] \times 8\mathrm{bytes}\,, \tag{34}$$

$$\mathrm{mem}\left[\texttt{ResampWorkspace}\right] = \left[(2+3\mathcal{R})\,N^{\mathrm{FFT}}_{\mathrm{samp}} + 4N_{\mathrm{Fbin}}\right] \times 8\,\mathrm{bytes}\,, \tag{35}$$

$$\tag{36}$$

and the total (multi-segment, multi-detector) "internal" memory usage of the Resampling $\mathcal{F}$-statistic calculation is therefore

$$\mathrm{mem}\left[\texttt{Resamp}\right] = N_{\mathrm{seg}}\,N_{\mathrm{det}} \times \mathrm{mem}\left[\texttt{ResampDataPerDetPerSeg}\right] + \mathrm{mem}\left[\texttt{ResampWorkspace}\right] \,. \tag{37}$$

Note that this accounting does not including the "external" memory required to store the returned $\mathcal{F}$-statistic results from each call to `XLALComputeFstat()`, namely

$$\mathrm{mem}\left[\mathcal{F}\texttt{-stats return}\right] = N_{\mathrm{Fbin}} \times \mathrm{mem}\left[\texttt{R4}\right] \,,$$
$$\mathrm{mem}\left[\texttt{per-IFO }\{\mathcal{F}^X\}\right] = N_{\mathrm{det}}\,N_{\mathrm{Fbin}} \times \mathrm{mem}\left[\texttt{R4}\right] \,,$$
$$\mathrm{mem}\left[\texttt{per-segment per-IFO }\{\mathcal{F}^{X,\ell}\}\right] = N_{\mathrm{seg}}\,N_{\mathrm{det}}\,N_{\mathrm{Fbin}} \times \mathrm{mem}\left[\texttt{R4}\right] \,,$$

depending on whether the user requested just return of $\mathcal{F}$-statistic, or per-detector $\mathcal{F}^X$, or also per-segment per-detector $\mathcal{F}^{X,\ell}$. This memory is handled by the calling code and has to be accounted for according to its usage of this memory (ie whether it is stored, or added to a toplist immediately, etc).

B. Demod Memory Model

The Demod code needs essentially no internal workspace other than the memory to hold the input SFTs, i.e.

$$\mathrm{mem}\left[\texttt{DemodDataPerSFT}\right] = (\Delta f_{\mathrm{load}}\,T_{\mathrm{sft}}) \times 8\,\mathrm{bytes}\,, \tag{38}$$

and so the total (multi-segment, multi-detector) Demod memory "internal" usage is

$$\mathrm{mem}\left[\texttt{Demod}\right] = N^{\mathrm{all}}_{\mathrm{sft}} \times \mathrm{mem}\left[\texttt{DemodDataPerSFT}\right] \,, \tag{39}$$

where $N^{\mathrm{all}}_{\mathrm{sft}}$ is the total number of all SFTs over all segments and detectors, i.e. $N^{\mathrm{all}}_{\mathrm{sft}} = N_{\mathrm{det}}\,N_{\mathrm{seg}}\,N_{\mathrm{sft}}$, if N_{sft} is the average number of SFTs per detector per segment.

III. TESTING THE TIMING MODEL

A. Timing on a Lenovo Thinkpad T520

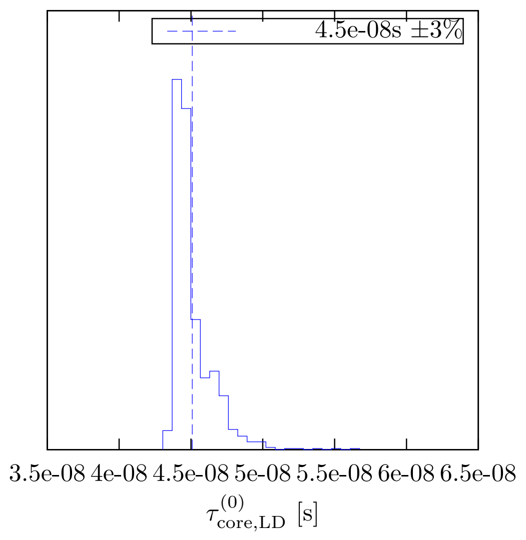
```
CPU: Intel(R) Core(TM) i7-2620M CPU, 2.70GHz,
L2 cache: 4MB
Extensions: sse sse2 ssse3 sse4_1 sse4_2 avx
Kernel: Linux 3.16.0-4-amd64
Compiler: Debian clang version 3.5.0-10 (tags/RELEASE_350/final) (based on LLVM 3.5.0)
```

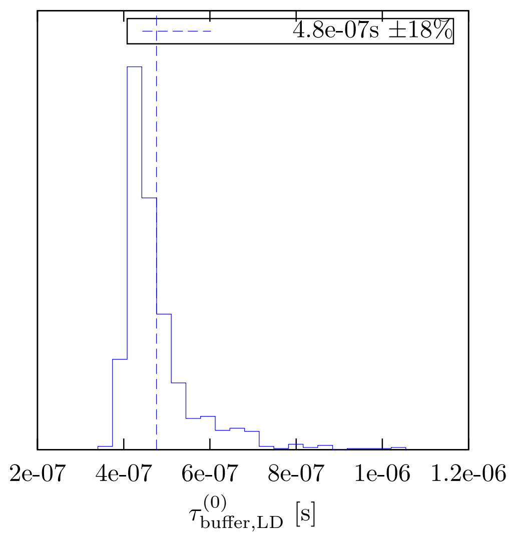
The timing measurements use `lalapps_ComputeFstatBenchmark`:

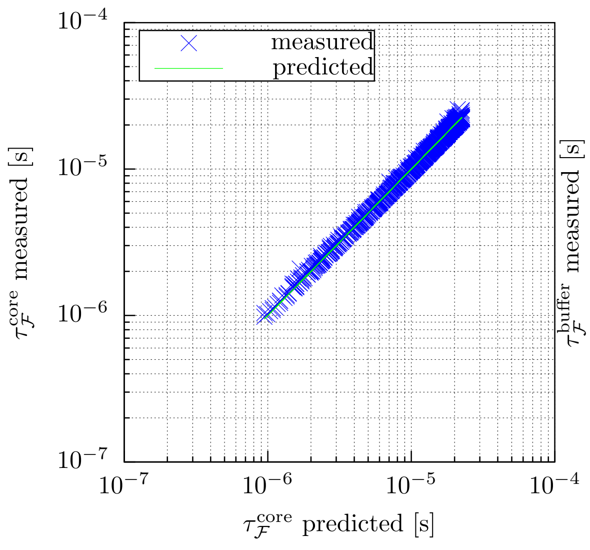
```
$ lalapps_ComputeFstatBenchmark --FstatMethod="ResampBest" --numSegments=1 --numTrials=1000
```

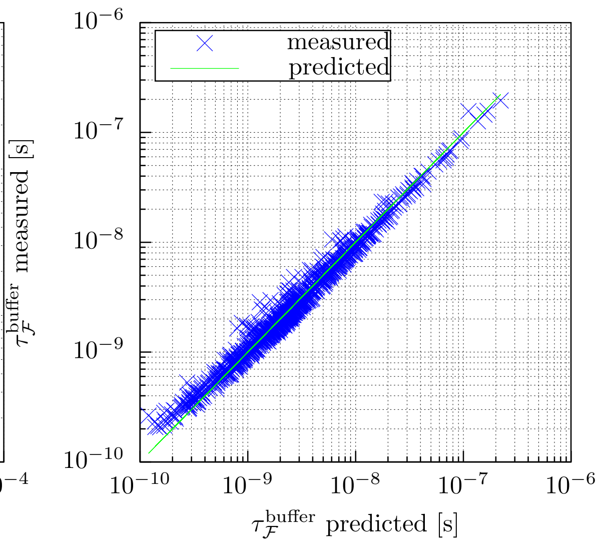
for Resampling, and `--FstatMethod="DemodBest"` for Demod. The model comparison was done using the octapps function `predictFstatTimeAndMemory()`.

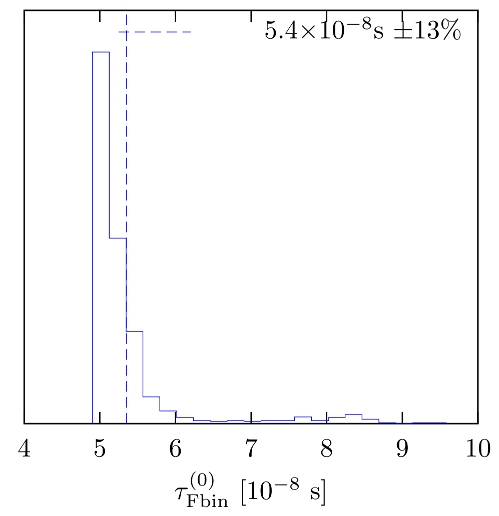
1. Demod timing: Thinkpad T520

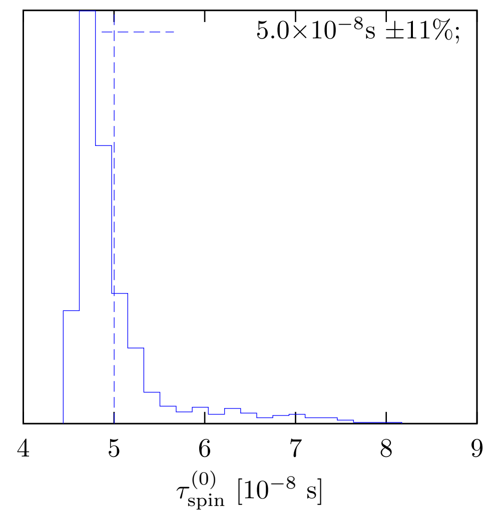


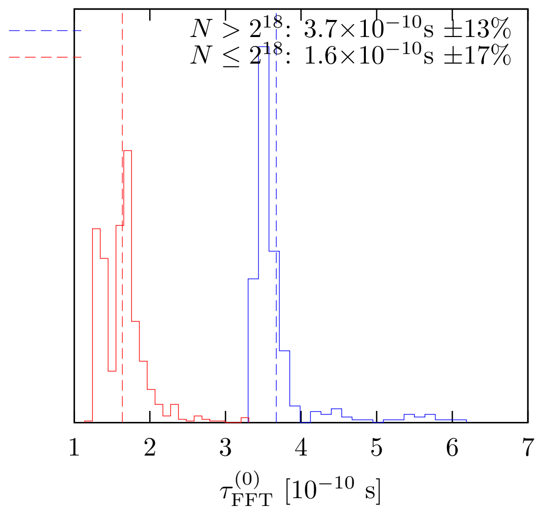


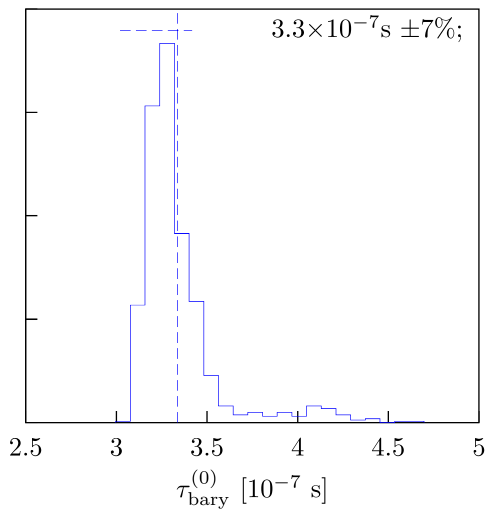


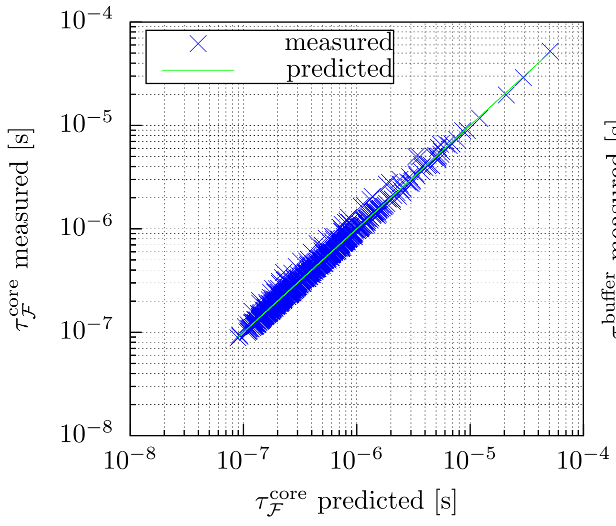


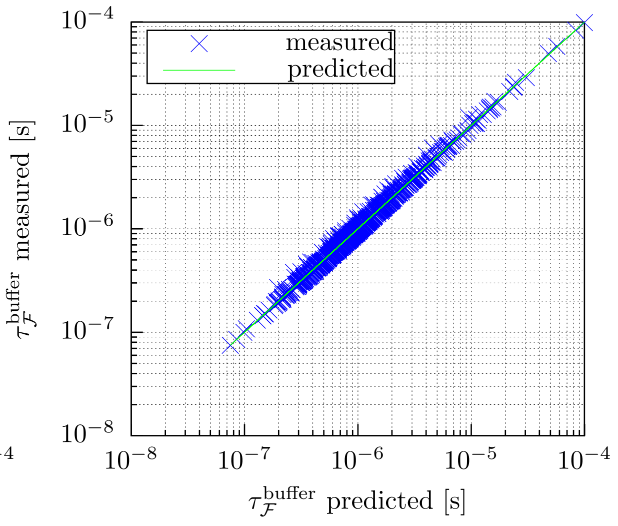

$5.4{\times}10^{-8}$s $\pm 13\%$
4
5
6
7
8
9
10
$\tau^{(0)}_{\mathrm{Fbin}}$ [10^{-8} s]


$5.0{\times}10^{-8}$s $\pm 11\%$;
4
5
6
7
8
9
$\tau^{(0)}_{\mathrm{spin}}$ [10^{-8} s]


$N > 2^{18}$: $3.7{\times}10^{-10}$s $\pm 13\%$
$N \leq 2^{18}$: $1.6{\times}10^{-10}$s $\pm 17\%$
1
2
3
4
5
6
7
$\tau^{(0)}_{\mathrm{FFT}}$ [10^{-10} s]


$3.3{\times}10^{-7}$s $\pm 7\%$;
2.5
3
3.5
4
4.5
5
$\tau^{(0)}_{\mathrm{bary}}$ [10^{-7} s]

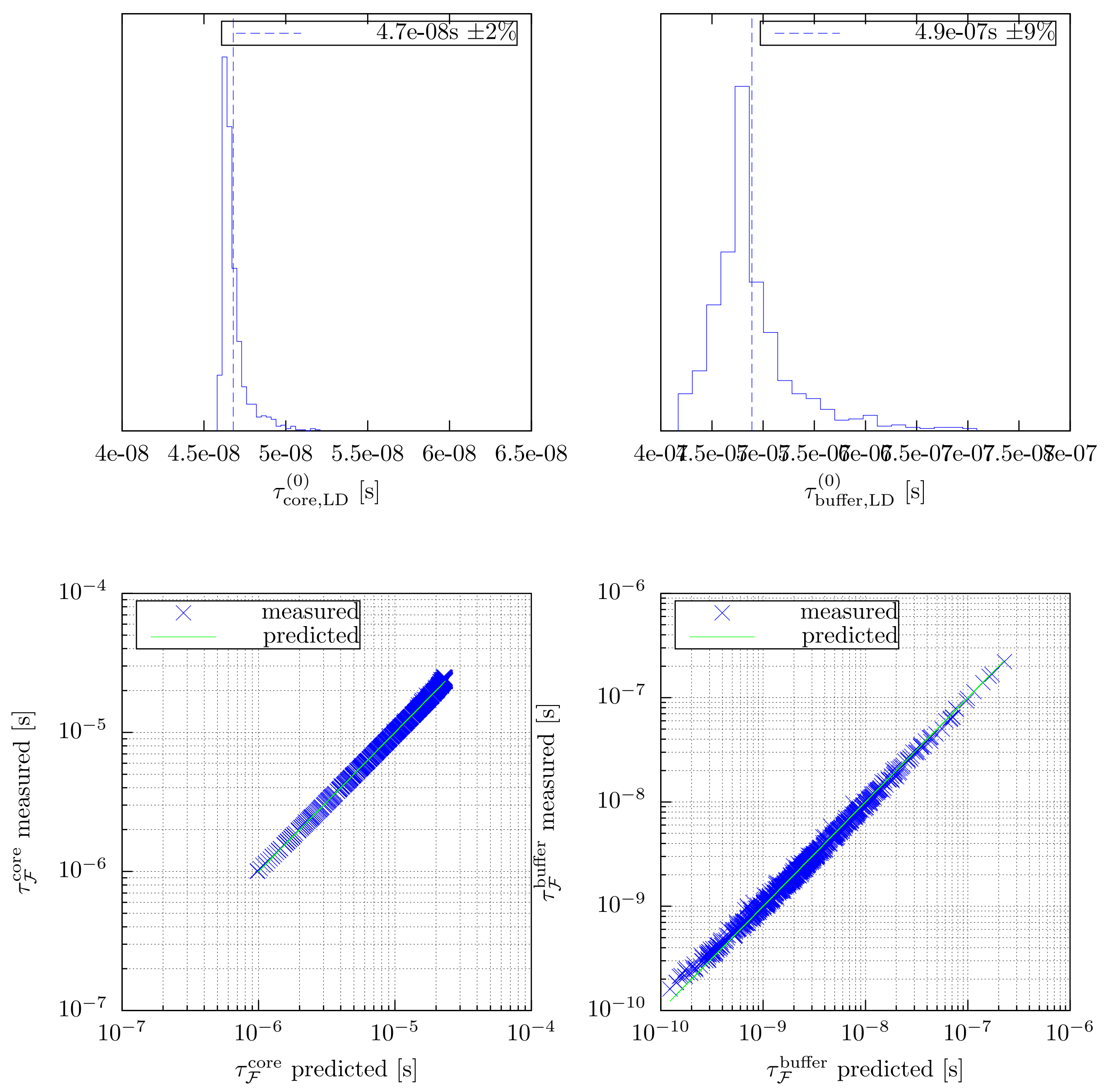

measured
predicted
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
$\tau^{\mathrm{core}}_{\mathcal{F}}$ measured [s]
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
$\tau^{\mathrm{core}}_{\mathcal{F}}$ predicted [s]

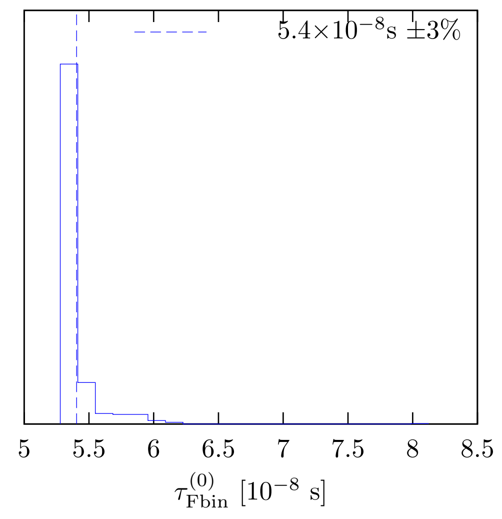

measured
predicted
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
$\tau^{\mathrm{buffer}}_{\mathcal{F}}$ measured [s]
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
$\tau^{\mathrm{buffer}}_{\mathcal{F}}$ predicted [s]

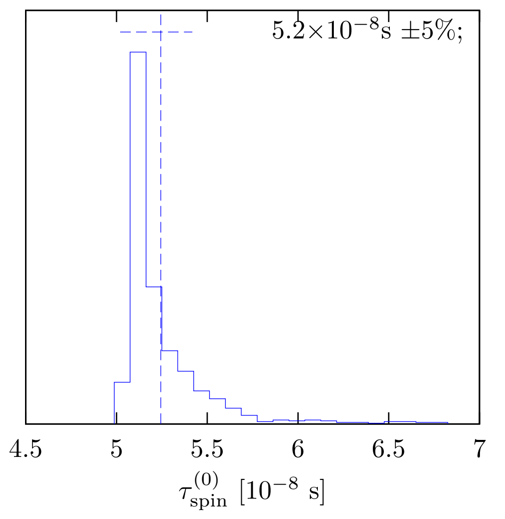
B. Timing on ATLASDEV1

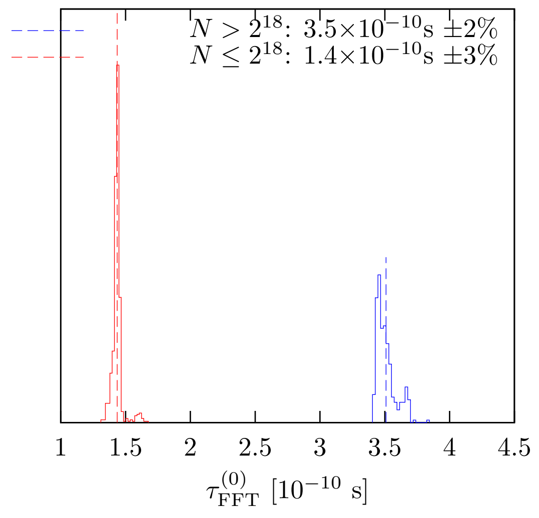
```
CPU: Intel(R) Xeon(R) CPU E5-1650 0 @ 3.20GHz
L2 cache: 12MB
Extensions: sse sse2 ssse3 sse4_1 sse4_2 avx
Kernel: Linux 3.16.0-4-amd64
Compiler: gcc (Debian 4.9.2-10) 4.9.2
```

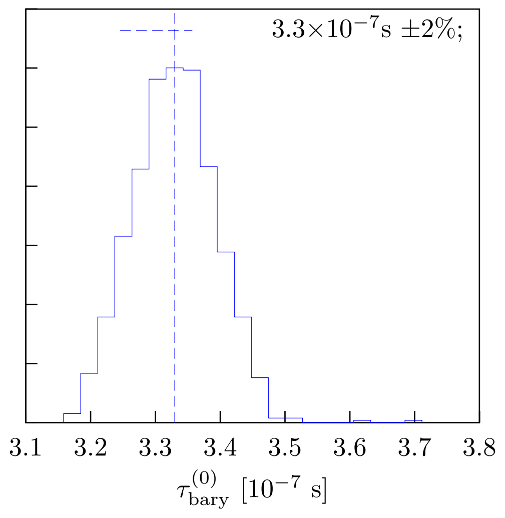
1. *Demod timing: ATLASDEV1*

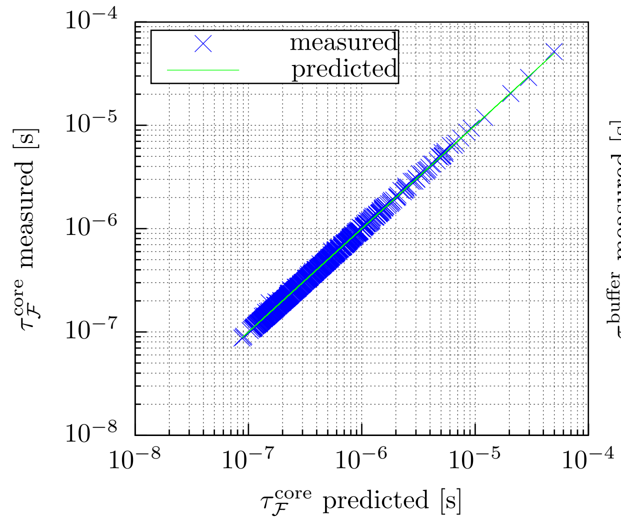


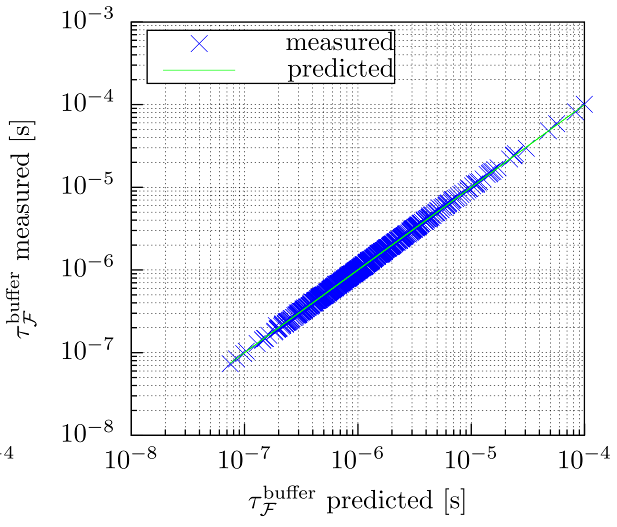

5.4×10⁻⁸s ±3%
5
5.5
6
6.5
7
7.5
8
8.5
$\tau^{(0)}_{\mathrm{Fbin}}$ [10⁻⁸ s]


5.2×10⁻⁸s ±5%;
4.5
5
5.5
6
6.5
7
$\tau^{(0)}_{\mathrm{spin}}$ [10⁻⁸ s]


$N > 2^{18}$: 3.5×10⁻¹⁰s ±2%
$N \leq 2^{18}$: 1.4×10⁻¹⁰s ±3%
1
1.5
2
2.5
3
3.5
4
4.5
$\tau^{(0)}_{\mathrm{FFT}}$ [10⁻¹⁰ s]


3.3×10⁻⁷s ±2%;
3.1
3.2
3.3
3.4
3.5
3.6
3.7
3.8
$\tau^{(0)}_{\mathrm{bary}}$ [10⁻⁷ s]


measured
predicted
10⁻⁴
10⁻⁵
10⁻⁶
10⁻⁷
10⁻⁸
$\tau^{\mathrm{core}}_{\mathcal{F}}$ measured [s]
$\tau^{\mathrm{core}}_{\mathcal{F}}$ predicted [s]


measured
predicted
10⁻³
10⁻⁴
10⁻⁵
10⁻⁶
10⁻⁷
10⁻⁸
$\tau^{\mathrm{buffer}}_{\mathcal{F}}$ measured [s]
$\tau^{\mathrm{buffer}}_{\mathcal{F}}$ predicted [s]

C. Average timing coefficients for E@H

As a reasonable "guess" for E@H timings, one could use the average timing coefficients. The following table summarizes the individual and average timing coefficients, all times are in seconds, and the FFT timing coefficients are for $N^{\rm FFT}_{\rm samp} \leq 2^{18}$ ("lo") and $N^{\rm FFT}_{\rm samp} > 2^{18}$ ("hi") respectively:

	Demod		Resampling			
Hardware	$\tau^{(0)}_{\rm core,LD}$	$\tau^{(0)}_{\rm bary,LD}$	$\tau^{(0)}_{\rm Fbin}$	$\tau^{(0)}_{\rm spin}$	$\tau^{(0)}_{\rm FFT}$ [lo,hi]	$\tau^{(0)}_{\rm bary}$
T520-new	4.5×10^{-8}	4.8×10^{-7}	5.4×10^{-8}	5.0×10^{-8}	$[1.6\times10^{-10}, 3.7\times10^{-10}]$	3.3×10^{-7}
atlasdev1-new	4.7×10^{-8}	4.9×10^{-7}	5.4×10^{-8}	5.2×10^{-8}	$[1.4\times10^{-10}, 3.5\times10^{-10}]$	3.3×10^{-7}
Average	4.6×10^{-8}	4.8×10^{-7}	5.4×10^{-8}	5.1×10^{-8}	$[1.5\times10^{-10}, 3.6\times10^{-10}]$	3.3×10^{-7}

[1] *LSC Algorithm Library - LALSuite*, FreeSoftware (GPL), URL `https://wiki.ligo.org/DASWG/LALSuite`.
[2] *Octapps - an LSC Octave library*, FreeSoftware (GPL), URL `https://gitlab.aei.uni-hannover.de/octapps/octapps`.